Amendments to the Claims:

This listing of claims will replace all prior versions, and listings, of claims in the application:

Listing of Claims:

1. (Currently Amended):

A method of loading a trustable operating system comprising:

identifying a region in a memory of a computer by a one of a plurality of processors;

loading a content into the identified region;

registering an identity of the content of the identified region, the registering comprises:

recording a hash digest of the content of the identified region, and

signing the hash digest with a hash signing engine having a secure channel to access the hash digest, the signed hash digest being stored in a register in the memory of the computer that is accessible to a third party to verify whether the content can be trusted; and

causing the one processor to jump to a known entry point in the content.

2. (Previously Presented):

The method of claim 1, further comprising:

preventing interference with the identifying, loading, and registering by at least one of a remaining one of the plurality of processors.

3. (Previously Presented):

The method of claim 2, wherein preventing interference comprises halting at least one of the remaining ones of the plurality of processors until the identifying, loading, and registering is complete.

4. (Previously Presented):

The method of claim 2, further comprising:

causing at least one of the remaining ones of the plurality of processors to jump to the known entry point in the content.

5. (Previously Presented):

The method of claim 1, wherein identifying comprises receiving a region parameter, the region parameter specifying a location of the region.

6. (Previously Presented):

The method of claim 5, wherein the location comprises a range of addresses in the memory of the computer within which the region is located.

7. (Original):

The method of claim 5, wherein the location comprises a start address and a length of the memory of the computer within which the region is located.

8. (Cancelled)

9. (Original):

The method of claim 1 wherein the content is a component of an operating system to operate the computer.

10. (Currently Amended):

The method of claim 9, wherein the component of the operating system is a one of ~~a Windows operating system, a Windows 95 operating system, a Windows 98 operating system, a Windows NT operating system, a Windows 2000 operating system,~~ a virtual machine monitor, and a privileged software nucleus.

11. (Original):

The method of claim 1 wherein identifying, loading and registering are uninterruptible.

12. (Currently Amended):

A article of manufacture comprising:

a machine-accessible medium including a data that, when accessed by a machine cause the machine to,

halt all but one of a plurality of central processing units (CPU) in a computer;

identify a region in a memory of the computer;

block access to the identified region by all resources except the non-halted CPU;

load a content into the identified region;

registering an identity of the content of the identified region, the registering comprises:

compute the cryptographic hash of the identified region,

~~record a~~ record the computed cryptographic hash of the content in the identified region, and

signing the computed cryptographic hash with a hash signing engine having a secure channel to access the cryptographic hash, the signed cryptographic hash being stored in a register in the memory of the computer that is accessible to a third party to verify whether the content can be trusted; and

cause the non-halted CPU to begin executing at a known entry point in the identified region.

13. (Original):

The article of manufacture of claim 12, wherein the data that causes the machine to halt the all but one of a plurality of CPUs comprises data causing the all but one of a plurality of CPUs to enter a halted state.

14. (Original):

The article of manufacture of claim 13, wherein the data further causes the halted CPUs to exit the halted state after the non-halted CPU has begun executing at the known entry point in the identified region

15. (Original):

The article of manufacture of claim 14, wherein the data further causes the previously halted CPUs to begin executing at the known entry point in the identified region upon exiting the halted state.

16. (Currently Amended):

The article of manufacture of claim 13, wherein the data that causes the machine to record the cryptographic hash includes data that further causes the machine to,

erase a hash digest area in the memory of the computer; and

record a required platform information ~~in the hash digest area;~~

~~compute the cryptographic hash of the identified region; and~~

~~record the computed cryptographic hash~~ in the hash digest area.

17. (Cancelled)

18. (Original):

The article of manufacture of claim 13, wherein the data that causes the machine to identify the region in memory of the computer includes data that further causes the machine to receive at least one region parameter containing a location of the identified region.

19. (Original):

The article of manufacture of claim 13, wherein the location includes an address of the identified region.

20. (Original):

The article of manufacture of claim 13, wherein the location includes a length of the identified region.

21. (Currently Amended):

A method of securing a region in a memory of a computer comprising:

halting all but one of a plurality of central processing units (CPU) in a computer;

blocking access to a region in a memory of the computer by all resources except the non-halted CPU;

registering an identity of content of the region of the memory, the registering comprises:

recording a cryptographic hash of the region;

signing the cryptographic hash with a digest signing engine coupled to the memory of the computer having a secure channel to access the cryptographic hash, the signed cryptographic hash being stored in a register in the memory of the computer that is accessible to a third party to verify whether the content can be trusted; and

placing the non-halted CPU into a known privileged state.

22. (Original):

The method of claim 21, further comprising causing the non-halted CPU to jump to a known entry point in the region.

23. (Original):

The method of claim 21, wherein halting comprises causing the all but one of a plurality of CPUs to enter a special halted state.

24. (Original):

The method of claim 23, further comprising causing the halted CPUs to exit the special halted state after the non-halted CPU has been placed into the known privileged state.

25. (Original):

The method of claim 24, further comprising causing the previously halted CPUs to begin executing at a known entry point in the region upon exiting the special halted state.

26. (Original):

The method of claim 21, wherein recording the cryptographic hash comprises:

erasing a hash digest area in the memory of the computer; and

recording a required platform information in the hash digest area;

computing the cryptographic hash of the region's contents; and

recording the computed cryptographic hash in the hash digest area.

27. (Original):

The method claim 26, wherein the hash digest area is a register in the memory of the computer.

28. (Cancelled)

29. (Original):

The method of claim 21, wherein the region is specified in at least one region parameter.

30. (Original):

The method of claim 29, wherein the at least one region parameter is an address of the region in the memory of the computer that is to be secured.

31. (Original):

The method of claim 29, wherein the at least one region parameter is a length of the region in the memory of the computer that is to be secured.

32. (Currently Amended):

An apparatus to load a trustable operating system comprising:

a first processor having a start secure operation (SSO), the SSO having a memory region parameter, wherein the first processor is capable of executing the SSO to block access to a region of memory specified in the memory region parameter and to place a content in the specified region;

a hash digest, wherein the first processor further is capable of executing the SSO to erase a current content of the hash digest and to record in the hash digest a cryptographic hash of the content of the specified region;

a digest signing engine having a secure channel to access the hash digest, the digest signing engine signing the hash digest and storing the signed hash digest in a register in the memory of the computer that is accessible to a third party to verify whether the content can be trusted; and

wherein the first processor further is capable of executing the SSO to unblock access to the specified region and to jump to a known entry point in the content of the specified region.

33. (Previously Presented):

The apparatus of claim 32, further comprising:

a second processor, the second processor having a join secure operation (JSO), wherein the second processor is capable of executing the JSO to prevent the second processor from interfering with the first processor's execution of the SSO.

34. (Previously Presented):

The apparatus of claim 33, wherein the second processor is capable of commencing execution of the JSO when the first processor commences execution of the SSO.

35. (Previously Presented):

The apparatus of claim 33, wherein, to prevent the second processor from interfering with the first processor's execution of the SSO, the JSO is capable of causing the second processor to enter a halted state until the first processor's execution of the SSO is complete.

36. (Previously Presented):

The apparatus of claim 35, wherein the first processor is capable of executing the JSO to further cause the second processor to exit the halted state after the first processor's execution of the SSO is complete and to begin executing at the known entry point in the content of the specified region.

37. (Currently Amended):

The apparatus of claim 32, further comprising ~~a digest signing engine having a secure channel to access the hash digest,~~ the digest signing engine capable of computing the cryptographic hash of the content in the specified region in response to a request by the first processor executing the SSO.

38. (Original):

The apparatus of claim 32, wherein the hash digest is a register in a memory of the apparatus outside the specified region.

39. (Currently Amended):

A method of loading a trustable operating system comprising:

selecting an area in a memory accessible to a processor;

loading a data into the selected area;

registering an identity of the data loaded in the selected area;

recording a unique cryptographic function of the data loaded in the selected area;

signing the unique cryptographic function with a hash signing engine having a secure channel to access the unique cryptographic function, the signed unique cryptographic function being stored in a register in memory;

directing the processor to commence processing at an entry point in the selected area; and

preventing interruption of the selecting, loading, registering, recording, signing, and directing until they are completed.

40. (Previously Presented):

The method of claim 39, wherein preventing interruption comprises halting any other processors having access to the memory until the selecting, loading, and directing is complete.

41. (Previously Presented):

The method of claim 40, further comprising:

causing the other processors to commence processing at an entry point in the selected area.

42. (Previously Presented):

The method of claim 39, wherein selecting comprises receiving a parameter specifying a location of the area to be selected.

43. (Previously Presented):

The method of claim 42, wherein the location is a range of addresses in memory within which the area is located.

44. (Previously Presented):

The method of claim 42, wherein the location comprises a start address and a length of memory within which the area is located.

45. (Cancelled)

46. (Previously Presented):

The method of claim 39 wherein the data is a component of an operating system to operate a device in which the memory resides.

47. (Previously Presented):

The method of claim 46, wherein the operating system has a graphical user interface.